Open Source Opens Opportunities for Army's Simulation System

Douglas J. Parsons
Program Executive Office

Dr. Robert L. Wittman Jr.
MITRE Corporation

The One Semi-Automated Forces (OneSAF) Objective System is the U.S. Army's next-generation, entity-level, simulation system planned to provide a comprehensive set of tools supporting computer-based simulation event setup, execution, and review. Postured as an open-architecture, open-source application, the OneSAF program will put this software into the hands of a vast number of developers throughout the Department of Defense with the intent of creating unprecedented participation across the modeling and simulation community to include multi-service, international, industry, and academia experts in the evolution of the OneSAF system. This article describes the factors that led OneSAF to an open source development methodology, the open source principles OneSAF is supporting, and the key processes and tools supporting the open source development.

Since its beginnings in 1991 with a small group of programmers – some considered fanatics – Linux has become a force of hundreds of thousands [1]. The value of open source computing is found in the ability to leverage the talents and resources of an entire community. The Program Executive Office for Simulation, Training, and Instrumentation (PEO STRI) is positioning its next-generation constructive simulation to provide this same benefit to the Army's modeling and simulation (M&S) community.

The One Semi-Automated Forces (OneSAF) Objective System (OOS) is being developed to primarily serve training audiences, research scientists, and acquisition analysts. The OOS will also provide embedded simulation capabilities as part of the Army's Future Combat Systems. Once fielded in fiscal year 2006, the OOS will not only be used by the Army, but also will serve multi-service, international, industry, and academic organizations. Releasing source code to such a vast network of developers will certainly reap benefits for the Department of Defense (DoD) M&S community as a whole; however, distributing source code alone will not provide the optimal mechanism for a community to work together.

Initial efforts focused on developing a capable, robust, and extensible architecture supporting a toolkit that will allow users to grow the baseline. Active program office support, tools, and processes are also necessary to foster communication and increase the likelihood that community-developed capabilities will be integrated and shared with other users and developers. Finally, growing OOS product line capabilities will not be limited only to skillful Java programmers: A software toolset will allow a user who is not a programmer to build military entities, units, and their respective activities.

OneSAF Background
The OOS is the U.S. Army's next-generation simulation system that can represent a full range of military operations, systems, and control processes. It will accurately and effectively represent specific activities of combat; command, control, communications, computers, and intelligence; combat support; and combat service support. It is an entity-level simulation, meaning that it can simulate the activities of individual combatants or vehicles (as opposed to aggregate-level simulations, which represent combatants and vehicles as groupings). It will also provide the appropriate representations of the physical environment (e.g., terrain features, weather, illumination, etc.) and its effect on simulated activities and behaviors.

One aspect that makes the OOS unique among Army simulations is its design for use by three distinct Army M&S domains. Specifically, the Advanced Concepts and Requirements (ACR) domain uses M&S for experimentation and analyses on Army doctrine and force-related concepts. The Research, Development, and Acquisition (RDA) domain uses M&S for acquisition analyses focused on equipping and supporting currently fielded and future forces. Finally, the Training, Exercises, and Military Operations (TEMO) domain employs M&S to train the force. It does so using live simulation (actual equipment on training ranges), virtual simulation (immersing the trainee into a synthetic environment), and constructive simulation (war games using computer-generated forces).

It Is About Saving Resources
The OneSAF program concept originated in January 1996 following an extensive study concluding that the Army was caught in a wasteful spending cycle, making identical or similar enhancements to many legacy simulations across three different user domains. In May 1997, the deputy commanding general for Training and Doctrine Command approved the Mission Needs Statement for OneSAF, which stated:

The need for OneSAF capabilities is not a response to a specific warfighting threat against the force; the need is driven by the guidance to reduce duplication of M&S investments, foster interoperability and reuse across M&S domains, and meet the M&S requirements of the future force. [2]

The Army decided the best approach for overcoming the problems associated with the multitude of aging simulations was to create a single, general-purpose, entity-level simulation and associated simulation event support tool [3].

Lessons Learned From a Legacy Simulation
The OneSAF program has drawn many lessons from the now-retired Modular Semi-Automated Forces (ModSAF) program. While the ModSAF simulation was nowhere near providing OOS' required capabilities, it was an entity-level military simulation serving a multi-domain M&S community. If not for the decision to release the source code, ModSAF would likely have been relatively unknown.

Funded by the Defense Advanced Research Projects Agency in the early 1990s, ModSAF was developed to facilitate synthetic environment research in support of distributed interactive simulation applications. Word of the availability of source code quickly spread through the M&S community, and requests for the software steadily grew up to the point of its retirement in 2002. By its end, more than 200 organizations had placed requests for the source code. The OneSAF Program Office reaped many lessons learned from the ModSAF program – those worth continuing as well as those that needed improvement.

Two ModSAF characteristics deemed critical to the success of OneSAF include the release of source code and the responsibility to provide services to facilitate and

1. REPORT DATE **JAN 2005**	2. REPORT TYPE	3. DATES COVERED **00-00-2005 to 00-00-2005**
4. TITLE AND SUBTITLE **Open Source Opens Opportunities for Army's Simulation System**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Program Executive Office-Simulation Training and Instrumentation,(PEO-STRI),12350 Research Pkwy,Orlando,FL,32826**		8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)		10. SPONSOR/MONITOR'S ACRONYM(S)
		11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT **Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES **CROSSTALK The Journal of Defense Software Engineering, January 2005**

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT **Same as Report (SAR)**	18. NUMBER OF PAGES **4**	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**			

enhance communications among the OneSAF user community. How OOS embraces these characteristics is described in the following paragraphs.

Open Source and OneSAF

Releasing source code as part of DoD applications raises numerous questions ranging from security concerns to baseline configuration management to cooperative development and, finally, integration. To address these concerns and to abide by DoD acquisition guidelines, OneSAF necessarily qualifies its definition of open source development.

For the vast majority of organizations that will request the OneSAF baseline, the distribution process will be much like that employed with ModSAF where the program manager (PM) of OneSAF distributes the baseline with source code at no cost. This is a condition where OneSAF aligns directly with a primary tenet of open source software as defined by the Open Source Initiative; however, there are key distinctions between the open source tenets and the OneSAF distribution model. The Open Source Initiative defines open source as software that provides the following rights and obligations [4]:

a) No royalty or other fee imposed upon redistribution.
b) Availability of the source code.
c) Right to create modifications and derivative works.
d) May require modified versions to be distributed as the original version plus patches.
e) No discrimination against persons or groups.
f) No discrimination against fields of endeavor.
g) All rights granted must flow through to/with redistributed versions.
h) The license applies to the program as a whole and to each of its components.
i) The license must not restrict other software, thus permitting the distribution of open source and closed source software together.

Of these, a, b, c, g, and h apply to the OneSAF distribution process. While not classified, the OOS will have content (e.g., data, algorithms) deemed sensitive by the U.S. Department of the Army. The baseline cannot be freely distributed as defined for open source due to security reasons. As such, PM OneSAF must be selective in the distribution of the OOS baseline. Essentially, there are two basic commitments made when a user signs a OneSAF distribution agreement:

1. Authorization to redistribute the baseline is restricted to PM OneSAF.
2. Users who develop new functionality into the OneSAF baseline agree to provide those capabilities back to PM OneSAF for possible reintegration.

These constraints offer advantages across the OneSAF user community. Facilitating distribution through a single focal point allows the PM to have knowledge of whom and how users intend to use the baseline. This knowledge will enhance the ability to identify and integrate useful community-developed capabilities into future baseline releases.

Helping to Communicate

In light of the restrictions OneSAF imposes on pure open source distribution, the OneSAF leadership felt compelled to provide communication-enhancing tools and processes that were seen as critical to the

Table 1: *Enabling Tools for OneSAF Open Source Development*

Tool	Description
Web-Based OneSAF Objective System Development Portal	The cornerstone of the OneSAF development environment is <https://www.OneSAF.net>. It is a secure Web site that houses technical information and historical and current programmatic, organizational, and task order structure. Technical information from architectural designs down to the Application Programmer's Interface (API) descriptions can be found on the site. The API descriptions are provided by automated code scrappers that generate Javadocs on a periodic basis. User ID and password are required.
Apache HTTPS Server	The Apache server <www.apache.org> provides a Web server for the OneSAF.net environment.
Mailman	Distributed asynchronous discussions and archiving is provided via e-mail using the Gnu's Not Unix free, open source product Mailman <www.gnu.org/software/mailman>. Mailman provides an integrated Web environment for managing e-mail discussions and e-newsletter lists. It offers a complement of mail list functionality, including built-in archiving, automatic bounce processing, content filtering, digest delivery, spam filters, and Web-based list administration.
Concurrent Versioning System	Configuration management and revision control processes and services are built around the Concurrent Versioning System (CVS) <www.cvshome.org>. CVS version 1.10.8 is freely available open source software, and provides revision control for all software development and Web-based information developed and used by the OneSAF team.
The Dynamic Object-Oriented Requirements System	Automated support for requirements management and tracking is provided by the Dynamic Object-Oriented Requirements System (DOORS) <www.telelogic.com>. Although neither freely available nor open source, this automated tool supports the requirements-driven OneSAF development process. DOORS version 7.0 provides automated support for OneSAF's rigorous requirements analysis and tracking process and is accessible to all task order participants within the OneSAF Integration and Development Environment building. DOORS allows requirements storage and retrieval, and maintains linkages between user, system, and software requirements.
Together Control Center	Automated software design and development support is provided by the Together Control Center (TCC) version 6.0 <www.togethersoft.com>. The TCC is neither free nor open source, but is necessary to meet the managed Software Engineering Institute Level 4 software development process in use by the architecture and integration contractor. The TCC allows integrated access to a user-configurable suite of software development tools. These tools span the software development life cycle from analysis through test.
WebRT	Automated risk tracking, action-item tracking, and defect tracking are handled using the freely available open source WebRT tool <www.bestpractical.com/index.html>. WebRT 1.0.1-4 has been customized to provide a Web-enabled tool to track and manage defects, issues, risks, and action items within OneSAF.
Java	Java provides a platform-independent development language and development kit to OneSAF. Sun's Java version 2.0 <www.javasoft.com>, along with the Software Development Kit (SDK) version 1.4.1, provides the Java language programming foundation for the OneSAF integrated drive electronics. OneSAF is reviewing the capabilities and schedule for stepping up the next major release of the Java SDK.
XML Spy	As data architecture and management play a critical role across the pre-exercise, run time, and post-exercise activities, OneSAF is leveraging eXtensible Markup Language (XML) technologies including XML Spy <www.xmlspy.com>. XML Spy version 4.0 provides the OneSAF users within the IDE the ability to create XML schemas that comply with the OneSAF Data Interchange Formats (DIF) standards. XML Spy features a format checking and validation tool to cross-check a document against its DIF.

success of ModSAF. For OneSAF these tools leverage Web- and e-mail-based technologies. For a list and description of these tools and technologies, see Table 1.

These tools are actively in use and have provided huge dividends in terms of user engagement and feedback. As part of the normal OneSAF development process, users review demonstrable capabilities and code and then electronically submit their comments, enhancements, and/or changes with supporting documentation to the Web-served comment and defect repository. These submissions are reviewed, categorized, and assigned for action.

After OneSAF Full Operational Capability at the end of fiscal year 2005, these Web-based tools may be enhanced to support code updates that can be inserted into an integration branch, compiled, and automatically regression tested with the results posted to the OneSAF Web and notification e-mailed to the developer. Currently, architecture compliance tools and processes exist to allow external developers to plan for a specific level of integration with the OneSAF code. The external developers' decisions are dependent on their requirements and investment in existing applications. Prior to formal baseline integration, new code that has been successfully integrated will be posted for download at the *users own risk*.

A formal OneSAF Configuration Control Board (CCB) will choose which newly integrated capabilities to incorporate into the next formally released baseline. Once incorporated into the baseline, PM OneSAF assumes responsibility for these enhancements, distributes them within the normal baseline distribution process, and removes the pre-baselined code from the use-at-your-own-risk Web page.

In addition to sponsoring CCB meetings, OneSAF now holds regular user group meetings for both the domestic and international M&S communities. This user group meeting gives users the opportunity to exchange relevant information about One SAF and its individual programs, demonstrate new capabilities, voice concerns, raise issues, and make recommendations.

Filling the Gaps

It was also critical to OneSAF's success to improve several ModSAF architectural blemishes. These critical improvements include (1) providing a more composable and extensible software architecture; (2) focusing on and providing tools for non-programmers to extend the list of simulated entities, units, and behaviors; (3) providing mechanisms to support greater success when integrating user-developed code; and

(4) providing mechanisms to fully document the interfaces and code delivered.

To meet the challenges and limitations of earlier simulation systems, the OneSAF architects applied a software-based product-line architecture development approach. The product-line approach concentrates on identifying and defining interfaces between independent, architecture-level components, and then specifies how these existing components can be automatically composed into useful end-user applications. Looking back on the early – circa 2001– OneSAF architectural development, three key architecture-related tenets stand out as significant enablers to the four improvement areas mentioned above, and to the overall success of the program. These tenets include a coordinated architectural vision, an iterative and incremental development process, and close user and developer collaboration.

Tenet I: Create a Coordinated System Architectural Vision
For OneSAF, this was essential in orienting and maintaining forward momentum on two of the four critical improvement areas: overall architecture support to composability and extensibility requirements, and support for tool-based extensibility.

Composability and extensibility were viewed as essential for OneSAF because these characteristics were especially limited by ModSAF's architecture. ModSAF's aggregate applications made it difficult, expensive, and time consuming for the community to make specific independent modifications and for these modifications to be integrated back into the baseline. This was particularly true when multiple modifications were made to a single application.

The OneSAF architecture vision was developed early on in concert with the procurement team, the users, and the developers. The vision focused on system-level composability and the architecture's ability to support independent component development along well-defined interfaces within quality and functional compliance specifications. This was the key to enabling the government's task order procurement strategy of issuing multiple contracts to develop the independent system components.

An architecture and integration contract was also issued to finalize the interface, functional, and quality specifications, as well as to create the development infrastructure [3] and develop the initial toolset. By forging this vision early on, the program was able to develop a set of objective measures called *interface maturity levels* used to define objectives and emphasize and measure progress against the OneSAF requirement set.

Since program inception, from the

architecture level down to implementation, particular emphasis has been on the composability tools allowing end-users to compose new entities, units, and their associated behaviors from existing software primitives without the need to write or even access the source code. These tools are highlighted in the OneSAF architecture as the Model Composer tools and include the Entity Composer, the Unit Composer, and the Behavior Composer.

The Entity Composer allows a OneSAF simulation entity such as an aircraft, helicopter, tank, truck, or individual combatant to be composed from individual model components using a Windows-based graphical user interface (GUI). Model components may include visual or other radar-based sensors, specific types of weapons, specific communications devices, and specific mobility components such as *wheeled* or *tracked*. Additionally, the user can select specific types of physical models that regulate the vulnerability, load carrying capacity, and other physical aspects of the entity.

The Unit Composer supports grouping individual entities into military units and civilian organizations using a GUI-based front end. The Unit Composer allows visualization and modification of all entities and previously constructed units. Once a unit is constructed, specific behaviors defining the doctrine and tactics of the unit can be associated with the unit. These behaviors are constructed using Behavior Composer.

The Behavior Composer allows the graphical construction of entity and unit behaviors from existing primitives (software coded behaviors) and other composite behaviors. Composite behaviors are simply behaviors that are made up of other composite or primitive behaviors. The Behavior Composer allows the specification of sequential or parallel behaviors that provide the automated control, reactions, and overall behaviors of entities and units.

Tenet II: Use an Iterative and Incremental Development Process
For OneSAF, an iterative and incremental process enabled two important effects. First, it allowed the program to create and test, through successive iterations, a set of consistent, comprehensive, and useful architectural- and implementation-level documentation. It also allowed the program to test and streamline its support for external development and integration, again, by applying lessons learned through successive iterations of the integration and test process. Specifying the architectural vision on paper and then working toward that vision in code allowed negotiated changes to the architecture where necessary to maintain consisten-

cy with the code. Changes occurred due to developmental breakthroughs, clearer understanding of requirements, or necessary changes due to reuse of legacy applications.

OneSAF continues to use an eight-week build process as the cornerstone for software and system engineering activities. The builds begin with defined, measurable, and in many cases, demonstrable objectives and then progress through analysis, design, code, and unit test. During the subsequent build, the previous build's products are sent through a system-level integration and test process. At yearly intervals, the system is packaged and delivered to a restricted set of beta-test sites. OneSAF is currently in its Block C development cycle and has successfully released its Block A and B system products to more than 50 beta-test sites.

For OneSAF, the iterative and incremental process also enabled the drive to demonstrate capabilities early and often to support a rich level of user interaction and feedback across the ACR, RDA, and TEMO domains.

Tenet III: Establish and Maintain Close User and Developer Collaboration

For OneSAF, this enabled continual feedback, prioritization, and interpretation of key user needs and requirements. The current success of the OneSAF program is also attributed to these close developer and user interactions. From the start, the development and user representative teams have been collocated in the OneSAF Integrated Development Environment, thereby enabling easy and frequent interaction between the more than 100 system and software engineers, the government management and technical personnel, and the domain (ACR, RDA, and TEMO) user representatives. This interaction is key to shortening the development cycle, as domain-specific questions can be quickly resolved.

Conclusion

From the start, OneSAF leadership has encouraged non-traditional software development methods. Open source development remains central as a new way to create, distribute, proliferate, and extend the simulation capabilities promised by OneSAF. Using lessons from earlier simulations, open source concepts played heavily in the selection and creation of processes and tools to meet the requirements of the OneSAF program.

Open source development has already paid large dividends to the OneSAF program ranging from enhanced Web-based communications between developers and users gained from using open source Web-based servers, mail lists, and request tracking

software, to the advanced native capabilities within the Java development environment. OneSAF leadership believes the benefits of open source development will expand upon OneSAF's formal release by allowing a wide range of developers to contribute to and propose extensions back to the OneSAF baseline. Once integrated into the core baseline, a capability developed by a given organization is available for use and extension by the community of OneSAF users.

Although OneSAF is still in development, once formally released in fiscal year 2006, it will be available and distributed with source code free of charge to any organization within the DoD with a valid OneSAF requirement. While OneSAF is focused as an Army/DoD program, other inter-agency organizations (e.g., other services, homeland defense, emergency response groups/ police, etc.), industry, and academia can gain access to the application.

OneSAF will also be available to the international community. Typically, international arrangements are made either through data exchange agreements, foreign military sales cases, or project agreements. Because of the sensitive nature of OneSAF, removing the non-exportable data and algorithms will develop a specialized international baseline.

Finally, as OneSAF prepares to enter its final year of development prior to formal release, it will evaluate and grow the necessary capabilities to accommodate distributed open source development. It is expected, based on lessons learned and experience to date, that the necessary tool and process changes will be small incremental changes versus an explosive big bang event.

To find out more about the OOS, please contact the authors, PM OneSAF, Lt. Col. John "Buck" Surdu at <john.surdu@ peostri.army.mil>, or visit <www.one saf.org>.◆

References

1. Hasan, Ragib. "History of Linux." Vers. 2.1. University of Illinois at Urbana-Champaign, July 2002 <https://net files.uiuc.edu/rhasan/linux>.
2. Training and Doctrine Command. "Mission Needs Statement." U.S. Army, 2004 <http://onesaf.org/1SAFMNS. doc>.
3. Wittman, Robert L., and Anthony J. Courtemanche. "The OneSAF Product Line Architecture: An Overview of the Products and Process." SimTeCT 2002. Melbourne Convention Center, Melbourne, Australia, 13-16 May 2002.
4. Webbink, Mark H. "Understanding Open Source Software." Society for Computers and the Law (51) Mar. 2003 <www.nswscl.org.au/journal/51/ Mark_H_Webbink.html>.

About the Authors

Douglas J. Parsons is the lead engineer of the Intelligent Simulation Systems Team at the U.S. Army Program Executive Office for Simulation, Training, and Instrumentation. His primary focus is toward the successful development of the One Semi-Automated Forces Objective System. Parsons has a Bachelor of Science in mechanical engineering from North Dakota State University, a Master of Science in systems management from Florida Institute of Technology, and a Master of Science in industrial engineering from the University of Central Florida.

Program Executive Office-Simulation Training and Instrumentation (PEO-STRI)
12350 Research PKWY
Orlando, FL 32826
Phone: (407) 384-3821
E-mail: doug.parsons@peostri.army.mil

Robert L. Wittman Jr., Ph.D., currently works for the MITRE Corporation supporting the One Semi-Automated Forces Objective System program. He has been part of the U.S. Department of Defense modeling and simulation community since 1990. He has a Bachelor of Science in computer science from Washington State University, a Master of Science in software engineering from the University of West Florida, and a doctorate in industrial engineering from the University of Central Florida.

MITRE Corporation
3504 Lake Lynda DR
Orlando, FL 32817
Phone: (321) 235-7601
E-mail: rwittman@mitre.org